# グレブナ基底計算を効率的に行うための項順序自動設定法 (Computer Algebra : Algorithms, Implementations and Applications)

AUTHOR(S):

沢田, 浩之

数理解析研究所講究録 1295 巻 2002 年 171-177



# グレブナ基底計算を効率的に行うための項順序自動設定法

沢田 浩之 (Hiroyuki SAWADA) *
産業技術総合研究所
NATIONAL INSTITUTE OF ADVANCED INDUSTRIAL SCIENCE AND TECHNOLOGY

## 1 はじめに

グレブナ基底の計算において、用いられる項順序がその計算効率に大きな影響を持つことはよく知られている [4]。全次数逆辞書式順序 (degree reverse lexicographic ordering) は計算効率のよい項順序として知られているが [2]、常に効率的というわけではない。

本稿では、グレブナ基底を効率的に計算する項順序の自動設定方法を提案するとともに、その実験結果について報告する。なお、本手法で設定する項順序はブロック式順序 (block order)[2] とし、各ブロック内における項順序は全次数逆辞書式順序とする。

**記法 1 (ブロック式順序)**
$y_1,\ldots,y_\eta$ が上位、$z_1,\ldots,z_\zeta$ が下位のブロックに属し、各ブロックにおいて、「$i<j \Leftrightarrow y_i$ は $y_j$ よりも上位」および「$i<j \Leftrightarrow z_i$ は $z_j$ よりも上位」のとき、以下のように記す。

$$y_1 > \ldots > y_\eta >> z_1 > \ldots > z_\zeta$$

## 2 項順序設定法

与えられた多項式集合に含まれるある多項式 $f(x_1,\ldots,x_n)$ が $x_n - g(x_1,\ldots,x_{n-1})$ ($g$ は $x_1,\ldots,x_{n-1}$ の多項式) の形をしているとき、$x_n$ を他の変数よりも上位のブロックへ配置することが計算効率上望ましいということは、経験的によく知られている。本稿で提案する方法は、個々の入力多項式からこのような順序関係を導き出し、それをもとにして変数をブロックへ振り分け、その後各ブロック内における全次数逆辞書式順序を設定するものである。

項順序の設定は以下の手順で行われる。

1. 前処理
2. 変数のブロックへの振り分け
3. ブロック内順序の設定
4. 後処理

*h.sawada@aist.go.jp

## 2.1 前処理

前処理として、2変数線形多項式を逐次選び出し、それを用いて他の多項式に対する簡約化を施す。この操作は以下の意味を持つ。

> グレブナ基底の計算過程において、本質的に同一の順位を持つことになる変数が2つ以上存在する場合には、余分な変数を予め消去しておく。

与えられた多項式集合の中に、次式のような2変数線形多項式が含まれているとする。

$$ax + by + c. \tag{1}$$

ただし、$a$、$b$、$c$は定数である。また、仮に$x$の順位は$y$よりも上位であるものとする。グレブナ基底計算過程において行われる式(1)による簡約化は、本質的に、$x$に対して$-(b/a)y-(c/a)$を代入することに他ならない。すなわち、式(1)による簡約化によって$x$は$y$に置き換えられてしまうため、$x$に与えられた順位は以降の計算において意味を持たなくなる。

本質的に同一の順位を持つような複数の変数を残しておくことは、計算効率向上に有効な項順序を決定する上で混乱を引き起こす可能性があるため、このような簡約化が必要となる。

## 2.2 変数のブロックへの振り分け

$x_n - g(x_1,\ldots,x_{n-1})$なる多項式が与えられたとき、$x_n >> x_1,\ldots,x_{n-1}$とすることが計算効率上望ましいことは、経験的によく知られている。これは、以下のように拡張できる。

**経験則 1**

$x_i - g_i(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$ $(i=k,\ldots,n)$なる多項式が与えられたとき、$x_k,\ldots,x_n >> x_1,\ldots,x_{k-1}$とすることが、計算効率上望ましい。

また、経験則1を補完する意味で、経験則2をおく。

**経験則 2**

どのような$i$についても$x_i - g(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$とはならない多項式が与えられたとき、$x_1,\ldots,x_n$を同一のブロックへ配置することが、計算効率上望ましい。

ここで提案する方法では、まず、経験則1を用いて、変数ブロックの上位–下位関係を示す有向グラフを作成する。そして、このようにして作成された有向グラフを組み合わせることによって、多項式集合全体に関する有向グラフを作成し、これに示される上位–下位関係に基づいて変数が割り振られるブロックを決定する。そのようなブロックが一意に決定しない場合には、経験則2を適用することにより、ブロックを決定する。

変数のブロックへの割り振りは、以下の手順で行われる。

1. 有向グラフの作成
2. 閉路の解消
3. 各変数の割り振り先ブロックの決定

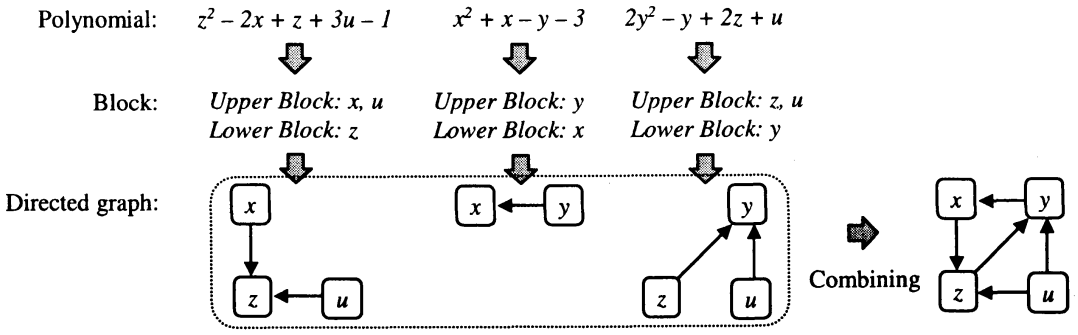


図 1: 有向グラフの作成

### 2.2.1 有向グラフの作成

変数ブロックの上位–下位関係を示す有向グラフの作成例を図 1 に示す。図 1 は、式 (2) に示す多項式集合 $F$ が与えられたときに作成される有向グラフを示している。

$$F = \{z^2 - 2x + z + 3u - 1, x^2 + x - y - 3, 2y^2 - y + 2z + u\}. \tag{2}$$

このように、経験則 1 に基づいて個々の多項式に関する有向グラフが作成され、それらを組み合わせることによって多項式集合全体に関する有向グラフが作成される。各変数が割り当てられるブロックは、この有向グラフで表される上下関係に基づいて決定される。

有向グラフが閉路を持つ場合、原則として、同一閉路上に存在する変数は同じブロックに割り振られる。経験則 1 に照らしてみた場合、この割り振り方は妥当なものだと考えられるが、2 つの変数ノードを直接結ぶ往復路に対しては必ずしもそうではない。以下に例を示す。

**2 つの変数ノードを直接結ぶ往復路が存在する場合における変数のブロック割り振りの例**

次式 (3) のような多項式が与えられているものとする。

$$x - f(z), y - g(z), z - h(x, y). \tag{3}$$

ただし、$f(z)$、$g(z)$、$h(x,y)$ は非線形多項式である。作成される有向グラフは図 2 となり、$x$、$y$、$z$ はす

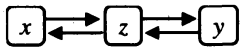

図 2: 2 つの変数ノードを直接結ぶ往復路を含む有向グラフの例

べて同一ブロックに割り振られることになる。だが、式 (3) の場合、$x$ と $y$ を $z$ よりも上位のブロックへ割り振れば、そのグレブナ基底は式 (4) となって直ちに求められる。

$$x - f(z), y - g(z), z - h(f(z), g(z)). \tag{4}$$

これは、$x$、$y$、$z$ を同一ブロックへ割り振った場合よりも、明らかに計算効率がよい。

したがって、2 つの変数ノードを直接結ぶ往復路が存在する場合には、いずれかの有向辺を削除しておくことが有効と考えられる。ここでは、各変数ノードに向かう有向辺の数を比較した結果によって、どちらかの有向辺を削除するという方法を採用する。

**定義 1 (往復路除去定数)**
変数のブロックへの割り振りに際し、1以上の定数である「往復路除去定数」を定義する。

2つの変数ノード $X$ と $Y$ を直接結ぶ往復路が存在するとき、次式 (5) が成立すれば、$X$ から $Y$ へ向かう有向辺は削除される。

$$\frac{(X\text{ へ向かう有向辺の数})}{(Y\text{ へ向かう有向辺の数})} > (\text{往復路除去定数}). \tag{5}$$

### 2.2.2 閉路の解消

作成された有向グラフに閉路が存在する場合、同一閉路上に存在する変数ノードは1つにまとめられる。閉路の解消は逐次行われ、閉路がすべてなくなるまで繰り返される。

### 2.2.3 各変数の割り振り先ブロックの決定

すでに閉路は存在しないので、この有向グラフは開始ノードと終端ノードを持つ。本稿で提案する方法では、開始ノードから終端ノードまでの最大距離を求め、それに基づいてブロックの数を決定し、各変数を割り振っていく。この操作について、図 3 を用いて説明する。

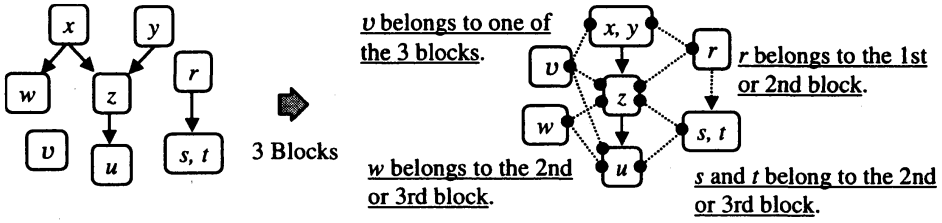


図 3: 各変数ノードの割り振り先ブロックの決定

まず、有向グラフの開始ノードをすべて選び出し、各開始ノードから終端ノードまでの最大距離を計算する。図 3 の場合、開始ノードは $\{x\}$、$\{y\}$、$\{r\}$ の 3 つであり、終端ノードまでの最大距離は 2 となる。これにより、ブロックの数は 3 と決定される。

次に、有向グラフで示される上位–下位関係と矛盾しないように、各ブロックへ変数ノードが割り振られる。図 3 の場合、$\{x\}$、$\{y\}$、$\{z\}$、$\{u\}$ の割り振り先ブロックは一意的に決定される。一方、他の 4 つの変数ノードについては、割り振り先が一意には決定しない。

これらの変数ノードについては、経験則 2 を適用することによって、その割り振り先ブロックを決定する。すなわち、各変数ノードについて、同一ブロックに割り振ることが好ましい変数のリストを作成し、そのリスト内の変数が占める割合が大きいブロックへ割り振る。

## 2.3 ブロック内順序の設定

ブロック内順序は、各変数の次数および出現回数に基づき、以下の手順で決定される。

**ブロック内の全次数逆辞書式順序決定手順**

1. 各変数について、与えられた多項式集合 $F$ における最大次数を求める。最大次数の小さい変数を上位にする。最大次数が等しい変数が複数存在する場合、ステップ 2 の規準に従って順序関係を決定する。

2. 各変数について、$F$に含まれる各多項式が持つ単項を、その変数の次数ごとにグループ分けする。そして、各グループに含まれる単項の数にその変数の次数を乗じたものの総和を計算する。この値が大きい変数を上位とする。
   この値は、変数の出現回数をその次数で重み付けしたものであるといえる。

### 2.4 後処理

第2.2節で述べたブロック決定法は、基本的に経験則1に基づいたものであり、経験則1は経験則2よりも優先することを暗黙のうちに仮定したものであった。しかしながら、経験則2に基づいて同一ブロックへの割り振りを示唆する多項式が多数存在するような場合には、これを優先させる方が合理的と考えられる。

ここで提案する後処理とは、前節までの手順で決定された項順序を、各多項式によって示唆されるブロック割り振りと照らし合わせ、その結果によって、隣接するブロックを1つのブロックに結合するものである。

**定義 2 (ブロック結合定数)**
隣接するブロックを結合すべきか否かを判定するために、「ブロック結合変数」を定義する。

隣接するブロック $B_1,\ldots,B_k$ に対し、式 *(6)* が成立するとき、これらは *1* つに結合される。

$$\frac{(B_1,\ldots,B_k \text{の結合を示唆する多項式の数})}{(B_1,\ldots,B_k \text{と矛盾しないブロックを示唆する多項式の数})} > (\text{ブロック結合変数}). \qquad (6)$$

なお、結合されたブロック内の変数の順序は、結合前の順序と同一とする。

## 3 評価実験

本項順序設定法の有効性を検証するため、以下の4つの例題を用いて評価実験を行った。

1. katsura-8 [1] (9 variables and 9 polynomimals)
2. Mckay [3] (4 variables and 20 polynomials)
3. Robot [6] (49 variables and 49 polynomials)
4. Heatpump [6] (21 variables and 20 polynomials)

評価は、本手法により生成された項順序によるグレブナ計算時間と、以下の2通りの項順序による計算時間を比較することによって行われた。

(a) ランダムに生成された全次数逆辞書式順序

(b) 本手法で生成したブロックのもとで、ブロック内順序をランダム生成した項順序

また、2つの制御パラメータ、往復路除去定数およびブロック結合定数の値は、ともに2とした。本実験でグレブナ基底計算に用いたソフトウェアにはRisa/Asir [5]であり、CPUはMobile Pentium-III 500MHz、RAMは320 MB、OSはWindows 2000である。

図4に実験結果を示す。Katsura-8とMcKayでは、本手法で生成されたブロックが1つだったため、ランダムに生成された全次数逆辞書式順序とのみ比較した。また、RobotとHeatpumpでは、10通りの全次数逆辞書式順序について計算したが、いずれもメモリ不足により計算が破綻した。そのため、ブロック内順序をランダム生成したものとのみ比較した。

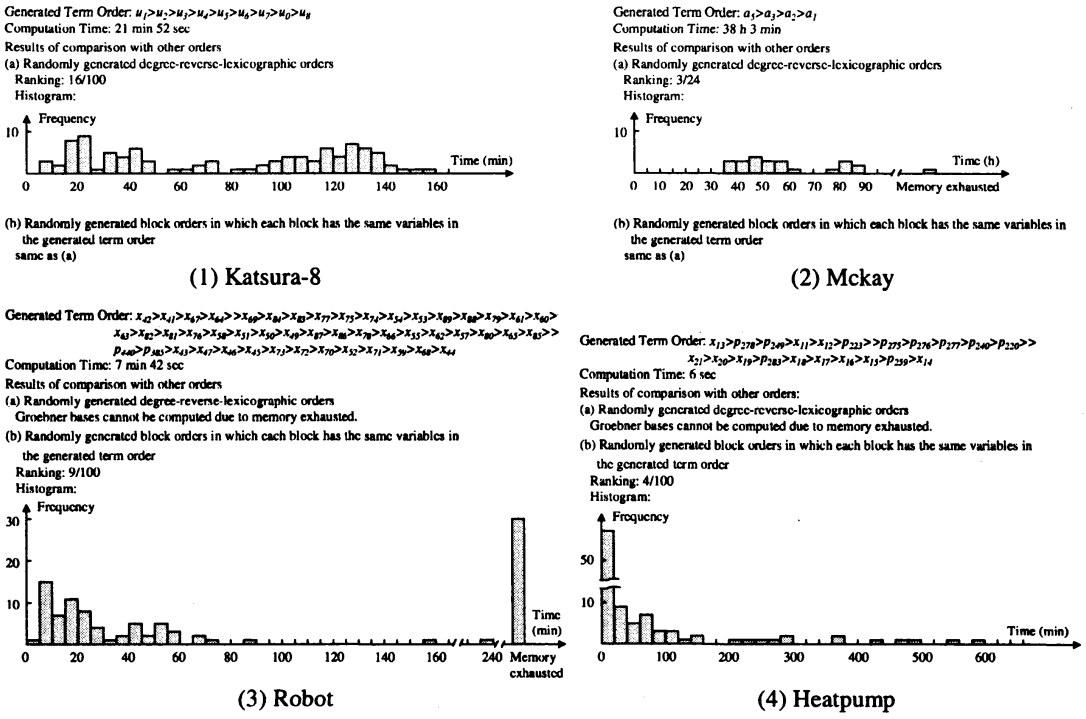


図 4: 実験結果

図 4 には、本手法で生成された項順序、その項順序におけるグレブナ基底計算時間、他の項順序と計算時間を比較した場合の順位、そして、計算時間のヒストグラムが示されている。ヒストグラムでは、横軸は計算時間、縦軸は度数を表している。これらの図から、本手法で生成された項順序が、グレブナ基底を効率的に計算する上で有利なものであることがわかる。

## 4 おわりに

グレブナ基底計算を効率的に行うためのブロック式順序の自動設定方法を提案した。本手法を 4 つの異なる例題に対して適用した結果、有意な差が認められ、本手法の有効性が検証された。また、実験により以下のことが例証された。

1. 全次数逆辞書式順序を用いたグレブナ基底計算は必ずしも効率的ではない。

2. ブロック内の変数順序も計算効率に大きくかかわる。

本手法により、計算アルゴリズムに関する知識を持たないユーザでも効率的にグレブナ基底を求められるようになる。これは、グレブナ基底の応用範囲を広げる上で大きな意味を持つ。

本稿で提案した手法は、すべて経験則に基づいたものであり、グレブナ基底計算を行うソフトウェアで採用しているアルゴリズムや実装方法に大きく依存している可能性は否定できない。今後、その部分も含めた有効性の検証を行う必要があるだろう。

# 参 考 文 献

[1] Katsura, S.: Theory of Spin Glass by the Method of the Distribution Function of an Effective Field, *Progress of Theoretical Physics*, Supplement, **87**, 1986, 139–154.

[2] 野呂正行: "グレブナ基底-理論, 計算の効率化, 応用", 数式処理における理論と応用の研究, 数理解析研究所講究録, **1138**, 京都大学数理解析研究所, 2000, 127–171.

[3] Noro, M., Mckay, J.: Computation of Replicable Functions on Risa/Asir, *Proc. PASCO'97*, ACM Press, 1997, 130–138.

[4] Ponder, C. G.: Evaluation of "Performance Enhancements" in Algebraic Manipulation Systems, PhD Thesis, Computer Science Division, University of California, USA, 1988.

[5] 齋藤友克, 竹島卓, 平野照比古: 日本で生まれた数式処理ソフト　リサアジールガイドブック, SEG 出版, 東京, 1998.

[6] Sawada, H.: Constraint-Based Computer Support for Insightful Multidisciplinary Engineering Design, PhD Thesis, University of Strathclyde, UK, 2001.